NESTAR SYSTEMS, INCORPORATED

CLUSTER/ONE MODEL A (tm)

```
*********************************************
*                                           *
*           Network File System             *
*                                           *
*           Disk block formats              *
*                                           *
*********************************************
```

Publication Number ZA20-0106-0

This document describes the organization of the disks managed by the Network File Server (NFS). This information is not necessary for normal use and maintenance of the NFS, but may be of use in special circumstances of disk error recovery. The reader is assumed to be familiar with the NFS to the level described in the System Manager's Manual (LC20-0102).

This information is accurate as of NFS version 1.3 (June 1982), and is subject to change.

NFS organization

It will be helpful to briefly describe the organization of the file server, so that the responsiblity for various parts of the disk organization can be shown.

The server software is layered according the the following scheme:

> Level 1. The actual disk access is performed by low-level "Disk Drivers", of which there is one per disk type (floppy, Century Winchester, etc.).
>
> Level 2. The "File System" uses the disk drivers and is responsible for allocating and freeing disk storage, maintaining the directory hierarchy, access control, and data buffering.
>
> Level 3. The "Queen" uses the file system (and the network drivers, not discussed here) to interpret and implement commands and data transfer requests from other stations on the network.

Most NFS offline utilities also are at level 3 and use the File System facilities.

```
    Queen            NFS utility (e.g. FILTRAN)

      |                   |
      |                   |
      V                   V

  File System        File System

      |                   |
      |                   |
      V                   V

 Disk drivers        Disk drivers
```

Disk block nomenclature

As far as the NFS and most file server utilities are concerned, all disks consist of a set of 256-byte blocks, numbered starting with zero and incrementing by 1 up to the number of blocks on the disk. All references to block locations on the disk are made using 4-byte block numbers.

The translation from 4-byte block number to physical disk addressing (cylinder number, head number, and sector number) is done by the low-level disk drivers just before accessing the disk. The details of that translation differs depending on the size (M20 vs M40), type (floppy vs Winchester) and manufacturer of the particular disk.

Note on terminology: The words "sector" and "block" are used interchangeably here to indicate a 256-byte chunk of data which is the smallest unit that can be read from or written to the disk. This is not to be confused with the use of the same words in the context of various user-station operating systems supported by the NFS, where the size may be different. For example, CP/M uses 128-byte sectors, and Pascal refers to 512-byte blocks. In specifying disk sizes, the NFS command language consistently uses S (for sector) to indicate a 256-byte allocation unit, and B (for block) to indicate a 512-byte allocation unit.

General Disk Organization

The general view of the disk is as a collection of files which are indexed using a multi-level hierarchical directory tree. Note the the "files" described here may themselves have an internal organization with inner files managed by a user-station operating system, in which case the file is sometimes called a "virtual disk" or "virtual volume".

There are 6 kinds of disk blocks that are stored on the disk:

1. The volume header, which describes the disk generally.

2. Freespace list blocks, which describe where the unused space on the disk is.

3. Descriptor blocks, each of which describes a single file or directory.

4. Directory blocks, which contain lists of other files.

5. User data blocks, which contain the data of a non-directory file.

6. Unused blocks, which are available for allocation.

Each of these blocks will be described in detail in the sections which follow. In addition, this document contains both Pascal-like declarations for the block formats, and examples of the blocks dumped in hexadecimal and ASCII. Finally, there is a diagram of the way in which the various disk blocks point to each other.

The first four block types (volume header, freespace lists blocks, descriptors, and directory blocks) begin with a 16-byte header part. That header part contains:

1. The characters "NSIB" in the first four positions.

2. The location of where this block is written on the disk.

3. The location of the logical predecessor ("father") of this block.

4. An indication of what type of block this is.

The information in the header is entirely redundant, and is used only to check that the disk file structures are healthy. During the execution of the NFS or any of the offline utilities, the discovery of incorrect header information will cause the current operation to be terminated and an INTERNAL ERROR to be reported. The FILCHECK utility will report in detail on disk structure errors.

The last two block formats (user data, and unused blocks) have no header information. Note, however, that it is possible for the arbitrary data in a user data block to resemble a header.

The Volume Header

There is only one volume header, and it is at block location 0. This is the ONLY block whose location is fixed; it must be at 0, the locations of all other blocks are known, directly or indirectly, from pointers in the volume header.

The volume header contains the name of the volume (which is also the name of the root directory), and pointers to the root directory and the start of the freespace list blocks.

The volume header also contains information about the size and structure of the disk. In the case where one software driver can access disks of different size, the header contains the driver-dependent structural information (like the number of tracks per cylinder) needed to access the rest of the disk.

Finally, the volume header contains a special pointer to the boot volume of the disk, so that the fileserver boot code need not be able to read and interpret the directory structure. See the section on "Disk blocks involved in booting the NFS" for more details.

The Freespace List blocks

The freespace list blocks describe the areas of the disk that are unused and available for allocation. Each block contains up to

29 free extent descriptor entries, and each such entry contains the location and size of a contiguous sequence of free blocks.

The freespace list blocks are created when the disk is formatted, and are linked together in a singly-linked list whose initial pointer is in the volume header block.

When an entry in a freespace list block is unused, the size described by that entry is zero. When the freespace list blocks are initialized, all the size fields are set to -1, which indicates that there are no used entries which follow; this is a "high-water mark" which allows the file system to ignore subsequent freespace list blocks and save disk accesses.

The File Descriptor Blocks

There is one descriptor block for every file on the disk, including directories. It contains a variety of information such as:

1. The file name and type.

2. The public, group, and private access rights.

3. The encoded group and private passwords.

4. The creation, last-access, and last-modified dates.

5. A description of up to 9 contiguous extents which contain the data blocks of the file.

The data blocks for the first extent of the file are normally allocated following the descriptor block for the file.

The Directory Blocks

When a file is a directory, the data blocks of the file are directory blocks. Each directory block contains a list of 10 filenames, along with the type and descriptor location of the file.

Unused entries in the directory resulting from file deletion are marked with type 'U'. When the directory is allocated or extended the directory blocks are zeroed, so that a type of ASCII 00 is used as a high-water mark to indicate that no valid file entries follow.

The User Data blocks

The data blocks of non-directory files are normally interpreted by user station programs and contain no data which is recognized by the file server.

In the case of virtual volumes (types P, D, C, 3, corresponding to Pascal, DOS, CP/M and SOS environments) and binary volumes (type B), the first data sector is used by the Queen to record special structuring information. For virtual volumes, the Queen may have to simulate accessing by track and sector, so information like the number of simulated sectors per track and the number of simulated tracks is recorded. For binary volumes, the load address and size in bytes is recorded. This first data sector is called the "Queen Descriptor".

Note, then, that for virtual volumes the first data sector accessed by the user station is the SECOND data sector of the file as far as the File System is concerned (sector #1, since the first sector is sector #0). Also remember that the local user station operating system may consider the primitive block size to be other than 256 bytes.

By NFS convention, the local operating system may request a 512-byte block numbered -1, which contains the File System descriptor in the first 256 bytes and the Queen Descriptor in the second 256 bytes. The -1 block may only be read, and the encrypted passwords are obliterated before being sent to the user station.

For example, UCSD Pascal expects 512-byte "blocks". The correspondence between File System sectors and Pascal blocks is thus

Purpose	File System Sector #	Pascal Block #
File System descriptor	---	-1, first half
Queen Descriptor	0	-1, second half
First data block	1 and 2	0
Second data block	3 and 4	1
etc.		

The Unused Blocks

Blocks which are unused, and are described in a freespace list entry, have no special format. Blocks which used to be either directory blocks or file descriptors are marked with "FREE" in the first four characters instead of "NSIB", so that scavenger utilities attempting to reconstruct damaged disks will not be confused with the remnants of deleted files.

Device-dependent mapping

The mapping from disk block number to physical disk addressing is done in a device-dependent manner by the disk drivers. In addition to information known privately, the drivers are given the 8-byte device information table ("DEVTAB") from the volume header to assist in the translation. For booting purposes, the DEVTAB supplied is zero, and the device driver is expected to be able to read block 0 without DEVTAB information.

For the Century Marksman M20 and M40 series 14" Winchester disk drives, only the first 4 bytes of DEVTAB are used:

Byte 0:	The number of heads per cylinder (4 for M20, 8 for M40)
Byte 1:	The number of sectors per track (76 for M20 and M40)
Bytes 2/3:	The number of sectors per cylinder (304 for M20, 608 for M40)

Two divisions thus suffice to produce the cylinder, head, and sector number from the block number. The first 76 sectors are track 0 head 0, the next 76 on track 0 head 1, and so on.

The Marksman driver also interleaves the sectors on the disk to produce an effective spacing of 20; the physical sector numbering on the track is sequential. The table used to determine the physical sector number from the logical sector number is as follows:

Logical sectors	Physical sectors
0..11	0,20,40,60,4,24,44,64,8,28,48,68
12..22	12,32,52,72,16,36,56,1,21,41,61
23..34	5,25,45,65,9,29,49,69,13,33,53,73
35..45	17,37,57,2,22,42,62,6,26,46,66
46..56	10,30,50,70,14,34,54,74,18,38,58
57..68	3,23,43,63,7,27,47,67,11,31,51,71
69..75	15,35,55,75,19,39,59

Using FILDEBUG

FILDEBUG is an offline NFS utility that can be used to examine and repair disk structures. FILDEBUG is often used in conjunction with FILCHECK; see the System Manager's Manual for a description of FILCHECK.

There are three major sections of the FILDEBUG utility which may be chosen by the displayed menu:

1. The D(isk) section allows disk blocks to be read into or written from any memory area. The sub-menu presented will ask for the disk unit number, the memory buffer address, the starting block location on the disk, the number of byte to transfer, and whether it is a disk read or disk write. There are defaults presented for all values; in particular note that the default for the memory address is that of an 4K buffer which may be used for temporary storage.

2. The M(emory) section allows an area of memory to be read (displayed) or written (modified). The sub-menu prompts for the the memory address and the length. The defaults for both are what were last specified for the D(isk) subsection buffer, to facilitate examination and modification of the last disk block read.

3. The F(ile system) section allows various File System commands to be issued. Some useful commands are:

```
 SETBOOT <pathname>  Sets boot volume pointer
 DUMPL <pathname>  Formatted dump of file information
 LIST  <pathname>  Like the NFS "list" command
 LISTN  <pathname>   Like  the  NFS  "list  ...,nested"
command
 INIT  Searches for all accessible disks
```

As an example of a sequence of FILDEBUG commands, one could modify the descriptor of a file by:

1. Using the DUMPL command of the F(ile system) section to find out where the file descriptor is located.

2. Using the D(isk) section to read the descriptor into the default buffer.

3. Using the M(emory) section to examine and modify the descriptor

4. Using the D(isk) section again to rewrite the descriptor.

Needless to say, any such disk block modification needs to be done with EXTREME caution to avoid damaging the disk structures.

Disk block format declarations

This section contains Pascal-like descriptions of the format of the various special file system sectors. These descriptions have had CONSTs and backward TYPE references removed to facilitate understanding of the mapping details, so that this is NOT the format of declarations used within NFS programs. These declarations are NOT legal pascal syntax.

The decimal numbers at the left edge are the offset of the field from the beginning of the block, and the size of the field in bytes. The format is "offset:size". In the case of repeated fields (arrays), only the size is shown as ":size".

The integers which represent disk locations or the sizes of disk files are 4-byte integers called BIGINTEGERs. All other integers are 2 bytes long.

Note that this implementation of Pascal stores integers backwards, that is, starting with the least significant byte and ending with the most significant byte.

Strings begin with a byte containing the current length of the string, followed by the characters of the string. Any characters from there to the end of the allocated length are irrelevant.

Special variable types

```
BIGINTEGER = 4-byte integer

        { used for disk addresses and file sizes }


DATE   = PACKED ARRAY[1..12] OF 0..15   { 6 bytes }

         { Date & time as YYMMDDhhmmss }


FOPNS = SET OF (RD,WR,AP,ER,CR,DE,FOP1,FOP2)   { 1 byte }

        { Legal file operations; one bit for each }
```

```
DBLK = RECORD { Every disk block is 256 bytes }

   { header part for all blocks, 16 bytes }

0:4    BID   : PACKED ARRAY [0..3] OF CHAR  { "NSIB" }
4:4    MYLOC : BIGINTEGER  { Self reference: location of this block }
8:4    FATHER: BIGINTEGER  { Back pointer: location of father }
12:2   spare : 2 BYTES

14:2   CASE BTYPE: INTEGER OF   { What type of block this is }

                 BTYPE=4 for volume header

                 BTYPE=3 for freespace list block

                 BTYPE=2 for file descriptor block

                 BTYPE=1 for directory block
```

```
 { Volume header block, BTYPE=4,  cyl 0, track 0, sector 0 }


16:16   UNITNAME:STRING[15]  { This disk's name }
32:2    VERSION :INTEGER     { Disk format version }
34:4    BOOTPGM :BIGINTEGER  { Loc of boot pgm }
38:4    DIRECTRY:BIGINTEGER  { Loc of root directory }
42:4    FREELIST:BIGINTEGER  { Loc of 1st freespace extent descr }
46:4    MAXBLKS :BIGINTEGER  { Max # of blocks on this logical disk }
50:4    FBLK    :BIGINTEGER  { 1st logical blk on this physical disk }
54:4    LBLK    :BIGINTEGER  { Lst logical blk on this physical disk }
58:8    DEVTAB  :PACKED ARRAY[0..7] OF 0..255
                            { Device characteristics table }
66:6    VLASTBACK:DATE       { Last volume backup date }
72:2    NALTTRK : INTEGER    { Number of alternate tracks assigned }
```

```
  { Freespace list block, BTYPE=3 }


16:4  NEXTBLK :BIGINTEGER      { Location of next block }
20:4  spare   :4 BYTES

24:232  FEXTENTS:ARRAY[1..29] OF RECORD {Free extent descriptor, 8 bytes }

:4      NBLKS  :BIGINTEGER  { number of blocks, -1 for HWM }
:4      EXTLOC :BIGINTEGER  { starting location of the extent }
```

```
{ Descriptor block, BTYPE=2 }

    { (1) File description }

16:80 DESCR:  PACKED RECORD      { FILE DESCRIPTOR, 80 bytes long }
16:16   FNAME    :STRING[15]     { File name }
32:1    FTYPE    :CHAR           { File types  FILETYPE
                          'Y'      Directory
                          'U'      Unused directory slot
                       CHR(0)      Neverused directory slot (HWM)
                          'x'      All others  }
33:1    spare    :1 BYTE         { ...Reserved for subtypes }
34:2    PRVPASS  :INTEGER        { Private password }
36:2    GRPPASS  :INTEGER        { Group password }
38:1    PUBACC   :FOPNS          { Public access rights }
39:1    PRVACC   :FOPNS          { Private access rights }
40:1    GRPACC   :FOPNS          { Group access rights }
41:1    LNKACC   :FOPNS          { Link access rights }
42:4    SIZE     :BIGINTEGER     { Allocated size in blocks }
46:4    LASTBLK  :BIGINTEGER     { Highest blk # written }
50:2    NLINKS   :INTEGER        { # of links to this file }
52:6    CREATION:DATE            { Date of creation }
58:6    LASTACC  :DATE           { Date of last access }
64:6    LASTMOD  :DATE           { Date of last modification }
70:6    LASTBACK:DATE            { Date of last backup }
76:32   spare     :32 BYTES      { ...Reserved. }

   { (2) File location information }

108:4   PEXTBLK :BIGINTEGER  { Location of next descriptor, if any }

112:144 PEXTENTS:PACKED ARRAY [1..9] OF  { up to 9 extent entries }

         EXTENTRY = PACKED RECORD  { Extent entry, 16 bytes each }
:4          FBLK      :BIGINTEGER  { 1st relative block # }
:4          LBLK      :BIGINTEGER  { last    "        "   " }
:4          EXTLOC    :BIGINTEGER  { Starting location, 0 for unused entry }
:2          spare     :2 BYTES
:1          LASTBYTE  :0..255      { # bytes in last block: not used }
:1          spare     :1 BYTE
```

```
{ Directory block, BTYPE=1 }


16:240  DIR: PACKED ARRAY[1..10] OF { 10 directory entries }

     DIRENTRY = PACKED RECORD  { each of which is 24 bytes long }
:16        FNAME :STRING[15]   { Name of the file }
:1         FTYPE :CHAR         { File type }
:1         LINK  :BOOLEAN      { Is this a link? }
:4         DESLOC:BIGINTEGER   { Ptr to descriptor block }
:2         spare :2 BYTES
```

```
{ Queen descriptor }
```

(The Queen descriptor is not managed by the File System and so does not have the 16-byte header.)

```
     QDESCR=RECORD
0:2     VOL:INTEGER  { DOS volume number        }
2:2     INIT:BOOLEAN { whether INITed by DOS    }
4:2     TSZ:INTEGER  { size of track in bytes   }
6:2     THI:INTEGER  { highest simulated track }
8:2     TLO:INTEGER  { lowest simulated track   }
10:2    SSZ:INTEGER  { size of sector in bytes }
12:2    SHI:INTEGER  { highest simulated sector}
14:2    SLO:INTEGER  { lowest simulated sector }
16:2    TYP:CHAR     { file type                }
18:4    BLKS:BIGINTEGER { # of 512 byte blocks for T=P and T=3
                          # of 256 byte sectrs other-ise }
22:2    LOADADDR: INTEGER { load address for binary files }
24:2    LEN : INTEGER     { length in bytes for binary files }
```

Examples of disk blocks in hex and ASCII

This section contains annotated dumps of representative samples of various disk blocks.

Volume Header

```
      length of name
          \
           \   NSIB      MYLOC=0     FATHER=0        BTYPE=4
  0/$0000 \: 4E53 4942 0000 0000 0000 0000 0000 0400 NSIB............
                  MAIN       unused part of name
 16/$0010  : 044D 4149 4E55 5000 0000 0000 0000 0000 .MAINUP.........
           VERSION BOOT VOL=01A151 ROOT DIR=1 FREELIST=5 MAXBLOCKS
 32/$0020  : 6400 51A1 0100 0100 0000 0500 0000 C0F2 d.Q.............
                    FBLK      LBLK      DEVTAB
 48/$0030  : 0100 0000 0000 8000 0000 084C 6002 0000 ...........L`...
                   LAST BACKUP
 64/$0040  : 0000 2860 5170 3180 0000 0000 0000 0000 ..(`Qpl.........
 80/$0050  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
 96/$0060  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
112/$0070  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
128/$0080  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
144/$0090  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
160/$00A0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
176/$00B0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
192/$00C0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
208/$00D0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
224/$00E0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
240/$00F0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
256/$0100  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
```

Root directory descriptor

```
             length of name
                  NSIB        MYLOC=1     FATHER=0         BTYPE=2
  0/$0000  : 4E53 4942 0100 0000 0000 0000 0000 0200 NSIB............
 16/$0010  : 044D 4149 4E55 5000 0000 0000 0000 0000 .MAINUP.........
           TYPE    PASSWORDS     RIGHTS      SIZE=3
 32/$0020  : 5900 26AA AAAA 0133 0000 0300 0000 0000 Y.&....3........
                         AA
                         CREATE DATE     ACCESS DATE
 48/$0030  : 0000 0000 1860 6022 3375 2860 6190 1115 .....``"3u(`a...
             MOD DATE
 64/$0040  : 2860 4090 3422 0000 0000 0000 0000 0000 (`@.4"..........
 80/$0050  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
 96/$0060  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
             FIRST BLOCK  LAST BLOCK  START LOC
112/$0070  : 0000 0000 0200 0000 0200 0000 0000 0000 ................  }
128/$0080  : 0000 0000 0000 0000 0000 0000 0000 0000 ................  }
144/$0090  : 0000 0000 0000 0000 0000 0000 0000 0000 ................  }
160/$00A0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................  }  9
176/$00B0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................  }  EXTENT
192/$00C0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................  }  ENTRIES
208/$00D0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................  }
224/$00E0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................  }
240/$00F0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................  }
```

Directory block of root directory

```
                                                                      DIRECTORY
                                                                      ENTRY
              NSIB       MYLOC=2    FATHER=1        BTYPE=1
  0/$0000 : 4E53 4942 0200 0000 0100 0000 0000 0100 NSIB............
 16/$0010 : 0653 5953 5445 4D00 0000 0000 0000 0000 .SYSTEM.........
 32/$0020 : 5900 1300 0000 0000 074F 4C44 424F 4F54 Y.......OLDBOOT
 48/$0030 : 0000 0000 0000 0000 4200 5FB5 0000 0000 ........B._.....
 64/$0040 : 0849 4E43 4F4D 494E 4700 0000 0000 0000 .INCOMING.......
 80/$0050 : 5900 68D1 0000 0000 084F 5554 474F 494E Y.h......OUTGOIN
 96/$0060 : 4700 0000 0000 0000 5900 75D4 0000 0000 G.......Y.u.....
name length
112/$0070 : 0343 504D 5042 4F4F 5400 0000 0000 0000 .CPMPBOOT.......
                      name    location of descriptor
TYPE
128/$0080 : 5900 8F20 0100 0000 0750 524F 4455 4354 Y.. .....PRODUCT
144/$0090 : 0000 0000 0000 0000 5900 D7D3 0000 0000 ........Y.......
160/$00A0 : 0555 5345 5253 0000 0000 0000 0000 0000 .USERS..........
176/$00B0 : 5900 8A73 0000 0000 0A4E 4653 2042 5241 Y..s.....NFS BRA
192/$00C0 : 484D 5300 0000 0000 5900 ECD7 0100 0000 HMS.....Y.......
208/$00D0 : 0753 4F55 5243 4553 4C00 0000 0000 0000 .SOURCESL.......
224/$00E0 : 5900 21D1 0000 0000 004B 424F 4F54 4F54 Y.!......KBOOTOT
240/$00F0 : 0000 0000 0000 0000 5500 0000 0000 0000 ........U.......

                                                      UNUSED
                                                      DIRECTORY
                                                      ENTRY
                                                      (T=U)
```

Freespace List block

```
   NEXT FREE
    LIST BLOCK
                   NSIB        MYLOC=5     FATHERPO         BTYPE=3
  0/$0000  : 4E53 4942 0500 0000 0000 0000 0000 0300 NSIB............
                                          SIZE      LOCATION
 16/$0010  : 0600 0000 0000 0000 0100 0000 1CD1 0000 ................
              SIZE      LOCATION  SIZE      LOCATION
 32/$0020  : C500 0000 74E8 0000 0100 0000 75E1 0100 ....t.......u...

 48/$0030  : 0200 0000 93D4 0100 0100 0000 D6D3 0000 ................

 64/$0040  : 0200 0000 EBA0 0100 0100 0000 C8EF 0000 ................

 80/$0050  : EE00 0000 FED6 0100 DA00 0000 6D36 0100 ............m6..

 96/$0060  : 0300 0000 2742 0100 0200 0000 07D4 0100 ....'B..........

112/$0070  : 1800 0000 3E0C 0100 0100 0000 750C 0100 ....>.......u...

128/$0080  : 1000 0000 7673 0000 0200 0000 EFD3 0100 ....vs..........

144/$0090  : 0200 0000 D743 0100 0100 0000 40A1 0000 .....C......@...

160/$00A0  : 0300 0000 500D 0100 0100 0000 3324 0000 ....P.......3$..

176/$00B0  : 0200 0000 F7D7 0100 0200 0000 8EF4 0000 ................

192/$00C0  : 0300 0000 4524 0000 7B00 0000 3ED6 0000 ....E$..{...>...

208/$00D0  : 0300 0000 35E5 0000 0200 0000 4FD4 0000 ....5.......O...
                                          SIZE      LOCATION
224/$00E0  : 0200 0000 B0EF 0100 0000 0000 320C 0100 ............2...

240/$00F0  : 0200 0000 FDA0 0100 A100 0000 AB2D 0000 .............-..

                                  AN UNUSED FREE EXTENT ENTRY
```

File descriptor (/MAIN/SYSTEM/BOOT)

```
                NSIB       MYLOC      FATHER          BTYPE=2
  0/$0000  : 4E53 4942 4FA1 0100 1300 0000 0000 0200 NSIBO...........
                  name
 16/$0010  : 0442 4F4F 5454 0000 0000 0000 0000 0000 .BOOTT..........
            TYPE=P PASSWORDS  RIGHTS    SIZE=$961
 32/$0020  : 5000 AAAA AAAA 010F 0101 6109 0000 0000 P.........a.....
                           CREATE DATE    ACCESS DATE
 48/$0030  : 0000 0000 1830 1230 0265 2860 6190 5225 .....0.0.e(`a.R%
              MOD DATE
 64/$0040  : 2860 6111 6040 1860 4181 2505 0000 0000 (`a.`@.`A.%.....
 80/$0050  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
 96/$0060  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
             FIRST BLOCK LAST BLOCK START LOC=$1A150
112/$0070  : 0000 0000 6009 0000 50A1 0100 0000 0000 ....`...P.......
128/$0080  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
144/$0090  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
160/$00A0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
176/$00B0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
192/$00C0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
208/$00D0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
224/$00E0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
240/$00F0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
```

Queen (Virtual volume) descriptor (T=P)

```
                   VOL=1  INIT                      SECTOR SIZE
256/$0100  : 0100 0100 0000 0000 0000 0001 0000 0000 ................
               TYPE     # of 512-BYTE BLOCKS
272/$0110  : 5000 B004 0000 0000 0000 0000 0000 0000 P...............
288/$0120  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
304/$0130  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
320/$0140  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
336/$0150  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
352/$0160  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
368/$0170  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
384/$0180  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
400/$0190  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
416/$01A0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
432/$01B0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
448/$01C0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
464/$01D0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
480/$01E0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
496/$01F0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
```

SECOND.BOOT

This is sector 1,2 (pascal block 0) of the boot volume.
For the example disk, it is block $01A151.

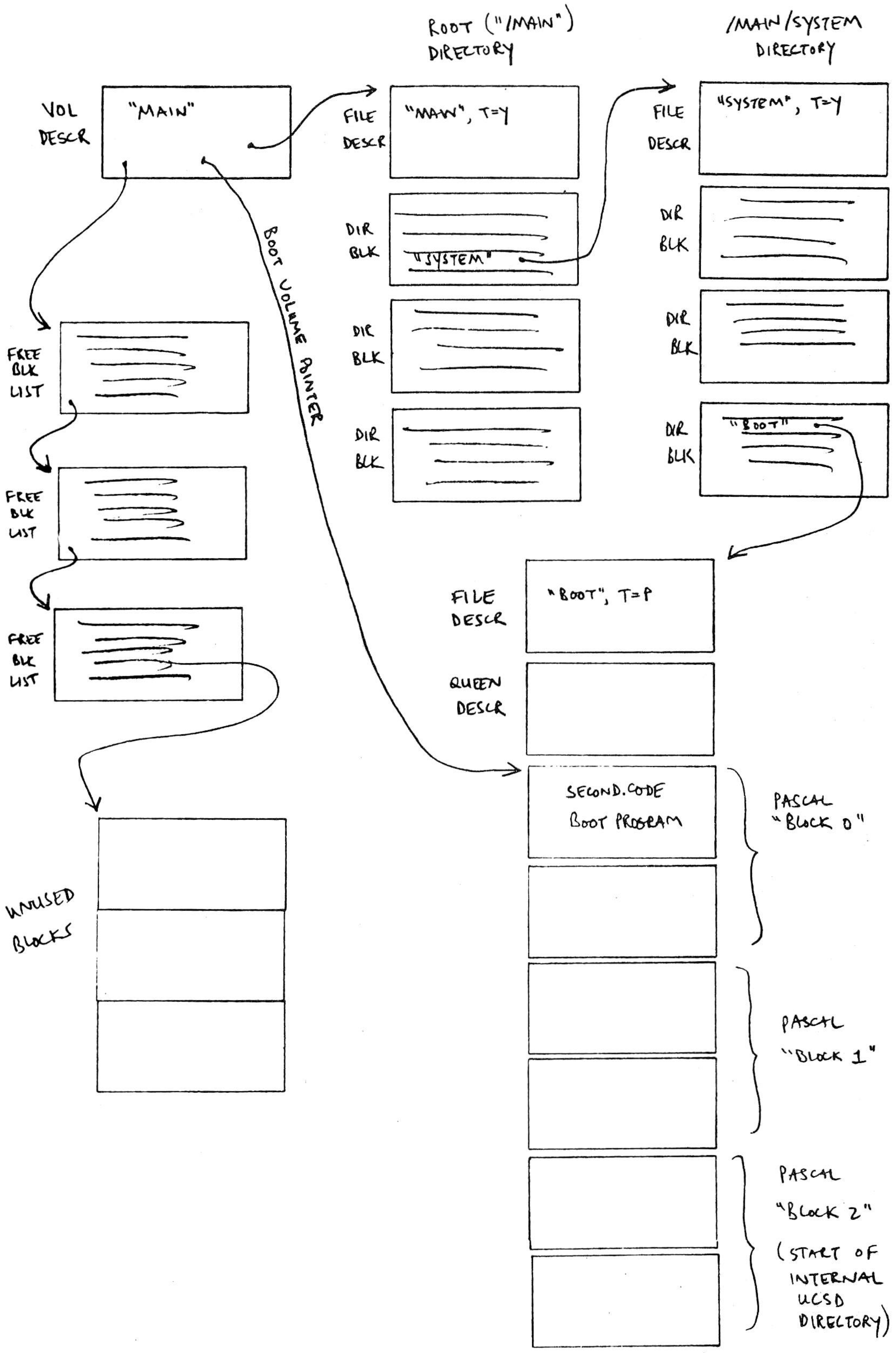
```
  0/$0000  : A508 8D2E 408D 7F40 8D3E 408D 3B40 8D41 ....@..@.>@.;@.A
 16/$0010  : 4020 5A40 AD06 41AC 0741 20A3 40B0 2020 @ Z@..A..A .@.
 32/$0020  : 8A40 A900 8D00 C8A9 5E8D 01C8 2080 C88D .@......^... ...
 48/$0030  : 09C8 D003 4C00 60A9 2120 83C8 4C89 C820 ....L.`.! ..L..
 64/$0040  : 86C8 4E4F 2046 494C 4520 5448 4952 442E ..NO FILE THIRD.
 80/$0050  : 424F 4F54 0D0A 004C 5740 A902 8518 A900 BOOT...LW@......
 96/$0060  : 8519 851A 851B 208A 40A9 008D 00C8 A930 ...... .@......0
112/$0070  : 8D01 C8A9 008D 02C8 A908 8D03 C820 80C8 ............. ..
128/$0080  : 8D09 C8D0 0160 A920 D0AF 1806 1826 1926 .....`. .....&.&
144/$0090  : 1A26 1BA0 FC18 B918 FF79 1CFF 9908 C7C8 .&.......y......
160/$00A0  : D0F4 6085 1284 13AD 0441 8510 AD05 4185 ..`......A....A.
176/$00B0  : 11A2 4EA0 00B1 12A8 B110 D112 DC06 8810 ..N.............
192/$00C0  : F74C D440 18A5 1069 1A85 1090 02E6 11CA .L.@...i........
208/$00D0  : D0E1 3860 38A5 10E9 0685 10B0 02C6 11A0 ..8`8...........
224/$00E0  : 00B1 1085 18C8 B110 8519 A900 851A 851B ................
240/$00F0  : 8D02 C8C8 38B1 10E5 188D 03C8 0E02 C82E ....8...........
256/$0100  : 03C8 1860 0630 0841 0A54 4849 5244 2E42 ...`.0.A.THIRD.B
272/$0110  : 4F4F 5400 0000 0000 0000 0000 1C01 0000 OOT.............
288/$0120  : 0200 0101 0000 0000 0000 0000 0000 0000 ................
304/$0130  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
320/$0140  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
336/$0150  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
352/$0160  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
368/$0170  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
384/$0180  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
400/$0190  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
416/$01A0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
432/$01B0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
448/$01C0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
464/$01D0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
480/$01E0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
496/$01F0  : 0000 0000 0000 0000 0000 0000 0000 0000 ................
```

Pascal directory (pascal blocks 2 to 5) of boot volume

```
  0/$0000  : 0000 0600 0000 0642 5241 484D 5300 B004 .......BRAHMS...
 16/$0010  : 1D00 0000 F3A3 0000 0000 0600 0A00 0200 ................
 32/$0020  : 0A54 4849 5244 2E42 4F4F 543D 0000 7667 .THIRD.BOOT=..vg
 48/$0030  : 0002 43A3 2200 2900 0200 0D43 4C4F 434B ..C.".)....CLOCK
 64/$0040  : 5345 542E 434F 4445 7667 0002 93A3 2900 SET.CODEvg....).
 80/$0050  : 5200 0200 0D53 5953 5445 4D2E 5041 5343 R....SYSTEM.PASC
 96/$0060  : 414C 7667 0002 69A1 5200 5300 0500 0F53 ALvg..i.R.S....S
112/$0070  : 5953 5445 4D2E 4D49 5343 494E 464F C000 YSTEM.MISCINFO..
128/$0080  : C1A2 5300 7300 0500 0C53 5953 5445 4D2E ..S.s....SYSTEM.
144/$0090  : 4150 504C 4500 7667 0002 9BA0 7300 8100 APPLE.vg....s...
160/$00A0  : 0200 0D44 4953 4B43 4F50 592E 434F 4445 ...DISKCOPY.CODE
176/$00B0  : 7667 0002 43A3 8100 8400 0500 0E53 5953 vg..C........SYS
192/$00C0  : 5445 4D2E 4C49 4252 4152 5967 0002 93A3 TEM.LIBRARYg....
208/$00D0  : 8400 8900 0200 0E53 5953 5445 4D2E 5354 .......SYSTEM.ST
224/$00E0  : 4152 5455 505E 0002 85A5 D400 DF00 0200 ARTUP^..........
240/$00F0  : 0A48 4449 534B 2E43 4F44 453D 0000 7667 .HDISK.CODE=..vg
256/$0100  : 0002 93A3 DF00 E400 0200 0D51 5354 4152 ...........QSTAR
272/$0110  : 5455 5031 2E31 2E31 5067 0002 43A3 E400 TUP1.1.1Pg..C...
288/$0120  : 0001 0200 0C53 5953 5445 4D2E 4649 4C45 .....SYSTEM.FILE
304/$0130  : 5200 7667 0002 29A1 0001 4901 0200 0B4E R.vg..)...I....N
320/$0140  : 4653 312E 312E 434F 4445 0000 7667 0002 FS1.1.CODE..vg..
336/$0150  : 15A3 4901 7001 0200 0B51 5345 5455 502E ..I.p....QSETUP.
352/$0160  : 434F 4445 0000 7667 0002 45A3 7001 A101 CODE..vg..E.p...
368/$0170  : 0200 0C46 494C 5452 414E 2E43 4F44 4500 ...FILTRAN.CODE.
384/$0180  : 7667 0002 45A3 A101 DA01 0200 0D46 494C vg..E........FIL
400/$0190  : 4445 4255 472E 434F 4445 7667 0002 45A3 DEBUG.CODEvg..E.
416/$01A0  : DA01 DE01 0300 0B4E 4653 434D 442E 5445 .......NFSCMD.TE
432/$01B0  : 5854 0000 E47B 0002 26A4 DE01 E001 0200 XT...{..&.......
448/$01C0  : 0E43 4841 494E 5445 5354 2E43 4F44 458E .CHAINTEST.CODE.
464/$01D0  : 0002 26A4 0302 2402 0200 0D54 4150 4544 ..&...$....TAPED
480/$01E0  : 554D 502E 434F 4445 7667 0002 27A2 3002 UMP.CODEvg..'.0.
496/$01F0  : 4902 0200 0C46 494C 4348 4543 4B2E 4F4C I....FILCHECK.OL
```

etc.

Diagram of Disk Block Relationships


ROOT ("/MAIN") DIRECTORY
/MAIN/SYSTEM DIRECTORY
VOL DESCR
"MAIN"
FILE DESCR
"MAIN", T=Y
FILE DESCR
"SYSTEM", T=Y
DIR BLK
"SYSTEM"
DIR BLK
DIR BLK
DIR BLK
DIR BLK
DIR BLK
"BOOT"
BOOT VOLUME POINTER
FREE BLK LIST
FREE BLK LIST
FREE BLK LIST
FILE DESCR
"BOOT", T=P
QUEEN DESCR
SECOND.CODE BOOT PROGRAM
PASCAL "BLOCK 0"
PASCAL "BLOCK 1"
PASCAL "BLOCK 2" (START OF INTERNAL UCSD DIRECTORY)
UNUSED BLOCKS